WELCOME.

Open the bloody presentation HAL!

I cant let you do that dave...

Class Rep

anyone want the post, with all the glory and infamy that goes with it?

COMPSCI 105

Summer School

Some rules

If you come in late, don't bother anyone

If you talk don't bother anyone.

Don't talk on your cellphone, or have it ring (use txts if you have to talk to people)

We will try to have a break halfway through the lecture.

Admin Stuff

Shane Paul (Course Supervisor)
Office: 303-376
Email:shane@cs.auckland.ac.nz

Daniel Bertinshaw (me)
Office: 303-576
Hours: 11-12 friday
Email:dber021@ec.auckland.ac.nz

Web:
www.cs.auckland.ac.nz/compsci105ssc
forums.cs.auckland.ac.nz
www.cecil.edu

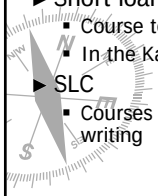
Text book
Data Abstraction and Problem Solving with Java
Carrano & Pritchard
we expect you to have and read this text!

About Assignments

- DON'T CHEAT
  - every assignment we catch at least three cheaters, this can result in no marks, or at worst, expulsion from the course.
- if your program doesn't compile, or has errors for simple cases: no marks
  - Always compile and check before submitting!
- You are welcome to challenge the marks… BUT!
  - if you think the marker was being cruel, or you don't understand why you got marked down: you will not get any more marks.
  - if you think you deserve more marks for the time you spent working: you will not get any more marks (we mark correctness).
  - if you "really need the marks to pass": you will not get any more marks (we have to be fair to everyone).
  - if you can tell me or Shane where the marker made a mistake, or missed something, please do tell us. we want accurate marks.

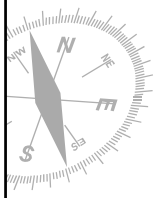
Resources

- Library
  - Books (no! really?)
  - Online information portals
  - Previous years exams
- Short loans
  - Course texts
  - In the Kate Edger commons
- SLC
  - Courses in stuff like time management and essay writing



TOPIC 1

Nested Loops
and
Multidimensional Arrays

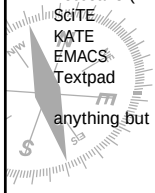


Software

Java 1.5.0
make sure you have this! we are going to use elements that are new to this version.
get the J2SE 1.5.0 R6 from java.sun.com

A decent editor
netbeans (written in java, good for forms)
SciTE (open source, i like it)
KATE (KDE only)
EMACS
Textpad

anything but notepad really...



Loops

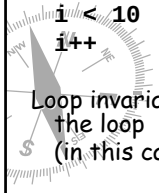
Recall our loops from 101

```
for (int i = 0; i < 10; i++)
```

Parts of the loop:

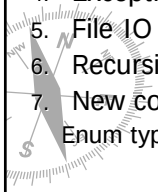
`int i = 0`	initializiation step
`i < 10`	condition
`i++`	increment step

Loop invariant: the value that doesn't change with the loop
(in this case: the value 10)



Topics I will cover

1. Multidimensional Arrays and Nested Loops
2. Software Design
3. ADT's and OO concepts
4. Exceptions and Flow Control
5. File IO
6. Recursion
7. New concepts in Java 1.5.0
   Enum types and Generics



top1eg1.java
top1eg2.java

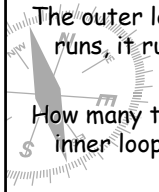
Nested Loops

Nested Loops are just loops inside other loops

```
for (int i = 0;i < 10; i++)
  for (int j = 0;j < 50; j++)
      //...
```

The outer loop runs 10 times, and each time it runs, it runs the inner loop 50 times

How many times in total does the code inside the inner loop get run?



Lots of things can be nested in computer science: classes, loops, branches, functions

Multidimensional Arrays

Arrays are reference types
Multidimensional arrays are arrays of arrays.

```
int [][] x = new int[3][3];
```

Creates a new 3x3 array of ints

`x` is an array of arrays of int
`x[0]` is an array of ints
`x[0][0]` is an int

top1eg4.java

Variable Sized Arrays

initialized arrays

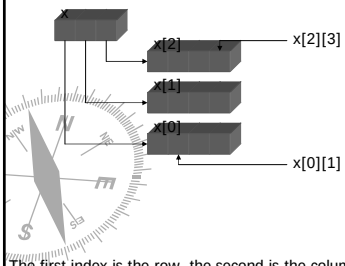
```
int[][] x = {{1,2,3},{5,6}}
```

or we can create each array using a new operator.

create the array inside the loop, and then loop over the newly created array.

Multidimensional Arrays

```
int [][] x = new int[3][4]
```



The first index is the row, the second is the column

When all the indexes are specified: the last object is an int, NOT a reference to an int

TOPIC 2

Software Design

This topic is pretty short, but its examinable.

top1eg3.java

Creating Arrays

initialized arrays:

```
int [] [] x = {{2,3,4},{4,5,6},{1,2,3}}
```

or by creating the array using the new operator and filling the values in with a nested loop

you need a loop for each dimension of the array

We can have as many dimensions as we want...

Designing Software

We need to make sure that we have the software mapped out before we dive into coding it.

This reduces time and effort
it seems like a lot of work, but it means that you know what you are dealing with, and when someone else looks at the code they can see what you were doing.

The SDLC

the **Software Development Life Cycle**

There are many versions, but the one in the book is:

1. Specification
2. Design
3. Risk Analysis
4. Verification
5. Coding
6. Testing
7. Refining
8. Production
9. Maintenance

top2eg1.psc

Design Schemes

For designing algorithms, pseudocode is generally used

a sequence of unambiguous english statements

there are a lot of different pseudocode definitions…

For designing large scale software projects, the dominant paradigm is Object Oriented Design.

The SDLC

We spend more time in the Design phase than anywhere else.

Get it right the first time…

Try to do this with your own work, even assignments.

Check out the textbook, pp 65 onwards

OO Design

We think of what objects or software is made of…

like the UI elements (buttons etc.), files, data (such as units in a game, or entries in a database)

We then define the relationships between objects.

Documentation

You should get into the habit of documenting everything you do.

Not just code, write out the design of the program, the files it uses etc.

It becomes a big issue for large projects…

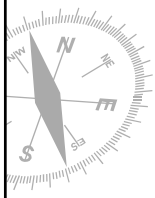
top7eg1.java

The ArrayList class

The arraylist class is much like an array BUT:
it self-manages size, so you don't run out of space.
it stores things as objects.

```
ArrayList a = new ArrayList ()
Object get(int)
   gets the object at the given index
Object remove(int)
   remove and return the object at the given index
void add(Object)
   add an object to the end of the list
void add(int, Object)
   add an object at the specified index
```

TOPIC 3



ADT's and OO Design

About abstracts

abstract methods do not have code.

we can override non-abstract methods, we cant override final methods.

although we called account.getValue, the program used either supplier.getValue or debtor.getValue.

we cant force it to get account.getvalue once it has been overloaded.

abstract: must be overridden
final: cannot be overridden
nothing: may or may not be overridden

Abstract Data Types

We sometimes notice things that are similar, an example from biology:

cephalopoda

nautilidea

nautilida 

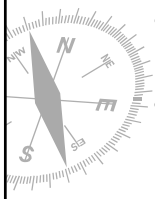
decapodiforms

sepiida 

tuethida 

octopodiforms

octopoda 

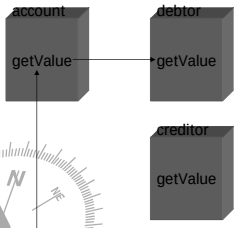


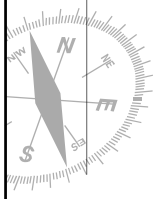
Notice that these are all similar

We have a kind of abstract view of them: the class "cephalopoda".

There are no animals that are just cephalopods, its an abstract concept grouping all animals with tentacles and no neck.

calling methods





when we call getValue control is passed to the class we created.

top3eg1/*.java

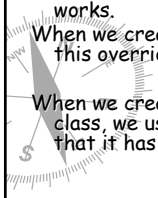
Creating abstract classes

when we create a class, we set some of the elements to be general ideas, using the keyword "abstract"

Abstract tells the compiler that the method must be overwritten to be useful. We cant do anything with an abtract method: it isnt code, just the idea of how code works.

When we create the new version of this method, we call this overriding.

When we create classes that have all the elements of the class, we use the "extends" keyword to tell the compiler that it has all the methods of the superclass.



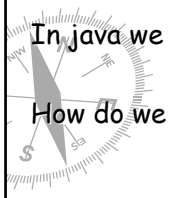
We must provide code for all abstract methods, not just some

Inheritance

we call this idea of making new classes out of old ones "inheritance" since they inherit the properties of the superclass.

In java we can only have one superclass…

How do we deal with wanting more?

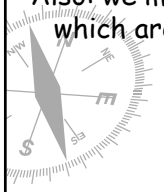


In C++ we can have multiple superclasses...

Some details on inheritance

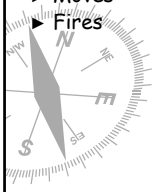
Every class in java inherits from java.lang.Object

Also: we may override methods in classes which are not abstract.



Using Interfaces

- Unit
  - Tank
  - Plane
  - Soldier
  - Building
- Moves
- Fires



to create a unit we might use

```
public class T1 extends
  Tank implements Moves,
  Fires
```

or

```
public class Howitzer
  extends Building
  implements Fires
```

you've already used interfaces with the event handlers in AWT/Swing

Interfaces

an interface is a collection of abstract methods

```
public interface x {
      public abstract int foo();
      public abstract double bar();
}
```

to use an interface, we specify them with the "impeplements" keyword

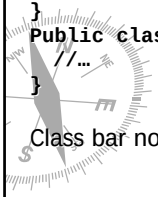
```
public class app extends jFrame implements
ActionListener, MouseListener
```

interfaces don't have data members

Inheritance and Interfaces

When a superclass impelments an interface, the subclasses also implement that interface

```
public class foo implements X {
  //…
}
Public class bar extends foo {
  //…
}
```

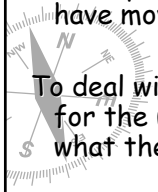


Class bar now implements X

Designing Interfaces

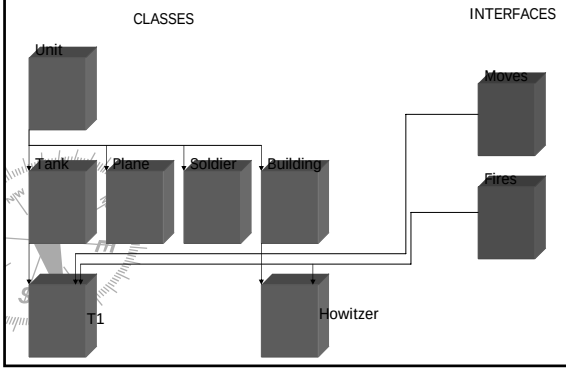
lets say we are making a game.

we have units, each unit has a type (soldier, tank, plane, building), but we also want to have moving units and firing units.



To deal with this, we make abstract classes for the units, and interfaces to describe what the units can do

a picture of our classes



MORE!

There is a lot more to OO design…

However, for this course, you need to understand inheritance and abstraction with overloading and interfaces.

Flow Control

The order in which instructions are processed.

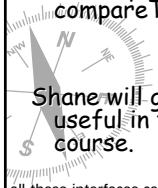
normally linear
- loops repeat instructions
- If branches instructions

but we have an extra method for flow control.

some useful interfaces

Clonable
- allows you to copy the object using the clone() method

Comparable
- allows you to compare two objects via the compareTo() method



Shane will go over more interfaces that are useful in the data structures portion of the course.

all these interfaces can be found in the java API documentation:
you REALLY should learn to use it ☺

top4eg1.java

Exceptions

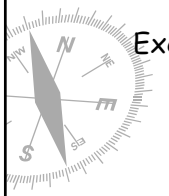
when an error occurs in java, the computer generates an "exception".

we have to tell the computer how to deal with the exception.

we do this with a "try-catch" statement.

TOPIC 4

Exceptions and Flow Control

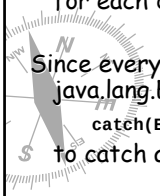


top4eg2.java

Multiple Exceptions

we can catch multiple exceptions for different possible errors

we do this by having a different catch statement for each different type of error.



Since every exception is a subclass of java.lang.Exception, we can use

```
catch(Exception e) {}
```

to catch all exceptions

top4eg2.java

finally blocks

a finally{} block is always executed, regardless of what the exceptions generated are.

we use it for dealing with various cleanup operations we might need.

if you use System.exit(), the finally block will not be executed!

defining new Exceptions

since exceptions are just classes, we can define new ones, by extending the class Exception

```
class StupidUserException extends Exception
```

If an Exception is a subclass of Exception, it must be caught. If it is a subclass of RuntimeException, then it does not need to be handled by the program

There is also a creature called an Error, but these are supposed to be so serious you shouldn't catch them

top4eg2.java

flow of exceptions

```
try {
    if (args.length != 2)
        throw new Exception("incorrect number of arguments");

    int q = Integer.parseInt(args[0]);
    int d = Integer.parseInt(args[1]);

    System.out.println(q/d);
}
catch (NumberFormatException e) {
    System.out.println("badly formatted number");
    return;
}
catch (ArithmeticException e) {
    System.out.println("cannot divide by zero");
    return;
}
catch (Exception e) {
    System.out.println(e);
    System.exit(1);
}
finally {
    System.out.println("thank you!");
}
```

leave the program

back to the calling method

some useful exception methods

printStackTrace
- prints the information about where the exception was generated

toString
- turns the exception into a string

getCause
- returns the cause of this exception

top4eg3.java

making someone else deal with it

we can declare methods to throw an exception, making the calling method have to deal with it.

```
public int foo (String args[])
   throws NullPointerException, ArrayIndexOutOfBoundsException{

    //...
}
```

now the method that calls foo() must either throw the exceptions, or have its calls to foo() in a try-catch-finally block

notice we use the keyword "throws"

we can use a try-catch-finally block, or make the method throw, NOT BOTH

Common Exceptions

NullPointerException
- when you try to access a class you havent instantiated

ArithmeticException
- general arithmetic error

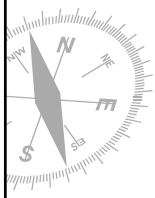
ArrayOutOfBoundsException
- trying to access an element of the array greater than the size or less than 0

ClassCastException
- trying to cast a class to a type that isnt a superclass or interface of the class

ClassNotFoundException
- trying to load a class and there is no .class file

TOPIC 5

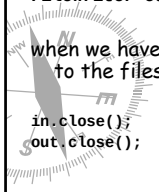
File IO



file reader/writer classes

when we create a new FileReader or FileWriter object, we open the file on the filesystem, or create it if it does not exist.

```
FileReader in = new FileReader (args[0]);
FileWriter out = new FileWriter (args[1]);
```



when we have finished with the file, we need to release it to the filesystem.

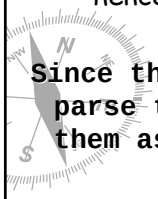
```
in.close();
out.close();
```

top5eg1.java

Command Line Arguments

when your program starts, everything you specify on the command line is put into the string array *args*

hence **`main(String args[])`**



Since they are strings, you must parse them if you want to treat them as ints, doubles etc…

you should get used to using the command line. by third stage you need to.

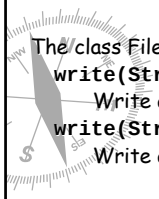
top5eg3.java

reading and writing

the class FileReader has the following methods we can use

`read(char[] cbuf)`
Read characters into an array.

`read(char[] cbuf, int off, int len)`
Read characters into a portion of an array.



The class FileWriter has the following methods:

`write(String str)`
Write a string.

`write(String str, int off, int len)`
Write a portion of a string.

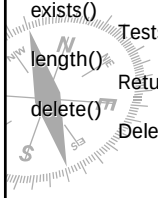
top5eg2.java

the File object

the file object represents something in the file system.

list()
Returns an array of strings naming the files and directories in the directory.

exists()
Tests whether the file or directory exists.



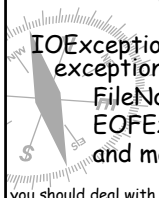
length()
Returns the length of the file.

delete()
Deletes the file or directory.

IO and Exceptions

most of the IO methods and classes make extensive use of IOExceptions.

you must use try-catch blocks to deal with them, since they are not runtime errors…

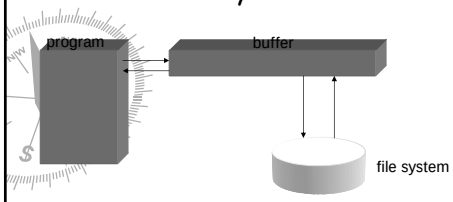


IOException is a superclass for a number of exceptions:
FileNotFoundException
EOFException
and many many many more…

you should deal with ones you might expect explicitly, and use IOException to catch everything else

Buffered IO

Buffering our IO streams makes them faster, by loading portions of the file into memory to be read, or storing portions of data in memory to be written later.



top5eg5.java

PrintWriter

PrintWriter allows us to print to files as if they were System.out

So we can use println() and print()

additionally, we could use the following methods

```
print (boolean b)
print (char c)
print (char[] s)
print (double d)
print (float f)
print (int i)
print (long l)
print (Object obj)
```

to print anything we want… ANYTHING!

methods for BufferedReader

`read()`
Read a single character.

`read(char[] cbuf, int off, int len)`
Read characters into a portion of an array.
both these methods return -1 at EOF

`String readLine()`
Read a line of text.
returns null at EOF

TOPIC 6

Recursion

methods for BufferedWriter

The methods for BufferedWriter are the same as FileWriter, so i don't really have much to say here!

`write(String str)`
Write a string.

`write(String str, int off, int len)`
Write a portion of a string.

As always: check the java documentation

Solving problems.

Lets say we have a list of numbers, and we want to add them:
[2 3 4 5 6]

Normally we would have a total and a current element. then we move through the list adding numbers…
total = 0 [2 3 4 5 6]
total = 2 [2 3 4 5 6]
total = 5 [2 3 4 5 6]
total = 9 [2 3 4 5 6]
total = 14 [2 3 4 5 6]
total = 20

top6eg1.java

solving problems recursively

lets come up with a different way of solving the same problem.

starting at the beginning, add the first two numbers, then repeat the idea with the new list.

[2 3 4 5 6]
[5 4 5 6]
[9 5 6]
[14 6]
[20]

When we have one element in the list, we stop.

This is called recursion.

Recursion in computer science

its not a loop. its not even like a loop.

we make methods that call themselves.

but we alter the arguments to simplify the problem.

and we have to stop it somehow.

as an interesting aside: Alonso Church's work (from the 1930's) shows that computer science is all recursive functions of binary strings.

Fibonacci numbers

the famous fibonacci numbers are a sequence
1, 1, 2, 3, 5, 8, 13…

The element at n is the sum of the element at n-1 and n-2.

Mathematically we can state this as:
f(n) = f(n-1) + f(n-2)
f(1) = f(2) = 1

MATHS! don't be fooled: compsci is a mathematical discipline.

parts of a recursive solution

every recursive function has two parts:

the recursive step (where we reduce the problem to a simpler one)

the stopping condition (when the solution is obvious, like f(0))

I like to call stopping condition "bailout conditions", from an older textboox

Recursion as an idea

when we use recursion, we take the problem, and solve a small part of it.

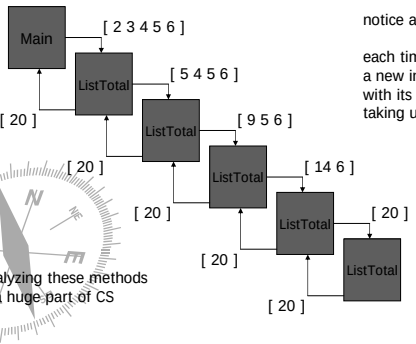
Creating a new, smaller problem.

then we take the new problem and apply our solution to that.

With our adding example, we are adding two numbers then using our algorithm to find the total of a smaller list.

dictionary entry:
Recursion *n* : see Recursion

what it 'looks' like



notice a problem?

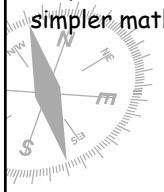
each time we call the method a new instance of it is created with its variables and arguments taking up memory.

analyzing these methods is a huge part of CS

top6eg2.java
top6eg3.java

pros and cons

Pros:	Cons:
simple to code	slower
simple to understand	takes more memory
simpler mathematically	



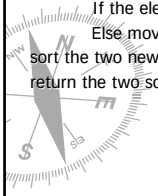
notice the number of method calls and nanoseconds taken for each program

do this with some cards…

Concept example 3

Sorting

pick an element in the list (call this a pivot).
  for all other elements in the list
    If the element is smaller, move it before the pivot
    Else move it after the pivot.
sort the two new lists.
return the two sorted lists concatenated with the pivot.



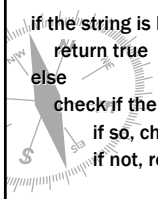
this is an algorithm called a quiksort: you will study it in depth later.

top6eg4.java

Concept example 1

a palindrome is a string that reads the same reversed
"able was i ere i saw elba"

if the string is length 1 or length 0
  return true
else
  check if the first and last characters are the same
    if so, check if the middle of the string is a palindrome
    if not, return false



a string of length 0 is called "the empty string"

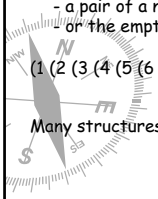
Recursive structures

Consider a list…
  (1,2,3,4,5,6)

what if we define a list as being made of lists?

a list is then:
  - a pair of a number and a list
  - or the empty list

(1 (2 (3 (4 (5 (6 ()))))))



Many structures are recursive, binary trees are a good example

The programming language LISP (made by John McCarthy in the 50's) is all recursive

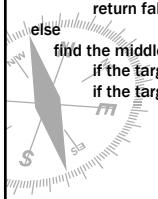
top6eg5.java

Concept example 2

searching

say we have a list of sorted numbers…

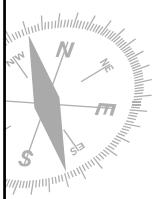
if we have a single element
    return true if the element is the one we want
    return false if not
else
  find the middle of the list,
    if the target is larger, find it in the larger part of the array
    if the target is smaller, find it in the smaller part of the array



This is a Binary Search

TOPIC 7

Generics and Enums
(new to Java 1.5.0)



Objects!

The ArrayList class works on objects, which means that you have to cast everything it returns (curses!).

So starting in java 1.5.0 there is a new method to deal with this. Generics.

now when we state the type eg ArrayList, we add in angle brackets what type this thing uses

ArrayList<Double>
ArrayList<String>

When you do data structures, remember that you can do this to most collections in java

enum types

Often we have to define a large number of constants

We do it like this:

```
public final int DEUCE = 2;
public final int THREE = 2;
public final int FOUR = 2;
…
public final int KING = 2;
public final int ACE = 2;
```

Painful process innit?

top7eg1.java
top7eg2.java

before and after

```
ArrayList a = new
  ArrayList ()
```

```
ArrayList<Double> a =
  new
  ArrayList<Double> ()
```

```
Object get(int)
Object remove(int)
void add(Object)
void add(int, Object)
```

```
Double get(int)
Double remove(int)
void add(Double)
void add(int, Double)
```

The doubles are how the funtion now acts…

top7eg3.java
Card.java

enum types

starting with java 1.5.0 we can define these constants using a type called an enum.

```
public enum Rank { DEUCE, THREE, FOUR, FIVE, SIX,
      SEVEN, EIGHT, NINE, TEN, JACK, QUEEN, KING, ACE }
```

To use an enum, we create a rank and give it a value from Rank.

```
Rank t = Rank.ACE;
```

The toString() method for enums always gives the type just as you made it

`t.tostring()` is `"ACE"`

so?

So now we don't have to cast obsessively.

Also, it will avoid a lot of ClassCastExceptions since we can only use one type in the collection.

and the following message goes away!

```
Note: top7eg1.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
```

Less code to write, less errors with generics.

Planet.java

enum constructors

enums have an implicit, private constructor.

This allows us to put data into an enum and treat it much like a class, but with pre-defined values.

when declaring the values, we put the arguments in brackets after the type:

```
HAT(1.20),
BELT(.70);
```

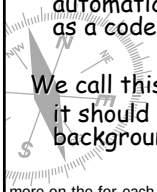
Check the example.

for-each

in the previous example, we had a for loop declared by

```
for (Planet p : Planet.values())
```

What this does is put each value into p, and will automatically handle exiting. making your life as a coder that much easier



We call this a for-each loop
it should be familiar to those with VB or C# backgrounds.

more on the for-each loop at:
http://java.sun.com/j2se/1.5.0/docs/guide/language/foreach.html

So long, and thanks for all the fish.

That's all from me, but ill still be tutoring.

Shane is going to take over for the rest of the course and teach data structures and efficiency